Model Optimization and Tuning Phase

Date	10^{th} July 2024
Team ID	SWTID1719999219
Project Title	Crystal Clear Vision: Revolutionizing Cataract Prediction through Transfer Learning Mastery
Maximum Marks	10 Marks

Model Optimization and Tuning Phase :

The Model Optimization and Tuning Phase involves refining neural network models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation :

Model	Tuned Hyperparameters
Resnet 50	Training with 5 Epochs - The model starts to learn basic patterns in the data with just 5 epochs, it’s highly likely that the model will underfit, meaning it hasn’t had enough time to learn the underlying patterns in the training data. The accuracy and loss metrics might show improvement, but the model’s performance will likely be suboptimal. Training with 50 Epochs - The model has more time to learn and adjust its weights, leading to better performance. With more epochs, there’s a risk of overfitting, where the model learns the noise in the training data, reducing its performance on unseen data. Using techniques like early stopping helps mitigate overfitting by stopping training once the model performance on the validation set stops improving. More epochs usually result in higher training accuracy and potentially higher validation accuracy if overfitting is controlled. Early Stopping: raining when the validation accuracy stops improving for a specified patience

Data Augmentation: Applying data augmentation as a layer (x=data_augmentation(inputs)) is a technique to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

```
#adding in data augmentation as a layer itself
x=data_augmentation(inputs)

#pass inputs to base model
x=base_model(x, training=False)
print(x.shape)

#We will use average pooling to reduce the size of the feature map.
x=tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
print(x.shape)

outputs=tf.keras.layers.Dense(1,activation='sigmoid',name='output_layer')(x)
model_0= tf.keras.Model(inputs,outputs)

#define early stopping callback.
early_stopping=EarlyStopping(monitor='val_accuracy',patience=3,restore_best_weights=True)
checkpoint = ModelCheckpoint('best_resnet_model.h5', monitor='val_accuracy', save_best_only=True)

model_0.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

#Using 50 epochs instead of 5
history_model_0= model_0.fit(train_data,
                             epochs=50,
                             validation_data=val_data,
                             callbacks=[early_stopping,checkpoint])
```

Accuracy –

Epochs = 5

```
Epoch 1/5
10/10 [==============================] - 52s 3s/step - loss: 0.5695 - accuracy: 0.7281 - val_loss: 0.5225 - val_accuracy: 0.7500
Epoch 2/5
10/10 [==============================] - 38s 2s/step - loss: 0.4380 - accuracy: 0.8156 - val_loss: 0.4202 - val_accuracy: 0.8000
Epoch 3/5
10/10 [==============================] - 32s 2s/step - loss: 0.3654 - accuracy: 0.8469 - val_loss: 0.3817 - val_accuracy: 0.8250
Epoch 4/5
10/10 [==============================] - 30s 1s/step - loss: 0.3323 - accuracy: 0.8687 - val_loss: 0.3507 - val_accuracy: 0.8750
Epoch 5/5
10/10 [==============================] - 39s 2s/step - loss: 0.3044 - accuracy: 0.8844 - val_loss: 0.3601 - val_accuracy: 0.8750
```

Epochs = 50

```
(None, None, None, 2048)
(None, 2048)
Epoch 1/50
10/10 [==============================] - ETA: 0s - loss: 0.4960 - accuracy: 0.7750/usr/local/lib/python3.10/dist-packages/keras/src/engine
  saving_api.save_model(
10/10 [==============================] - 51s 3s/step - loss: 0.4960 - accuracy: 0.7750 - val_loss: 0.4466 - val_accuracy: 0.8000
Epoch 2/50
10/10 [==============================] - 36s 2s/step - loss: 0.3761 - accuracy: 0.8219 - val_loss: 0.3689 - val_accuracy: 0.8250
Epoch 3/50
10/10 [==============================] - 37s 2s/step - loss: 0.3223 - accuracy: 0.8906 - val_loss: 0.3560 - val_accuracy: 0.8500
Epoch 4/50
10/10 [==============================] - 32s 2s/step - loss: 0.3046 - accuracy: 0.8781 - val_loss: 0.3775 - val_accuracy: 0.8500
Epoch 5/50
10/10 [==============================] - 37s 2s/step - loss: 0.2929 - accuracy: 0.8687 - val_loss: 0.3559 - val_accuracy: 0.8750
Epoch 6/50
10/10 [==============================] - 32s 1s/step - loss: 0.2997 - accuracy: 0.8719 - val_loss: 0.4078 - val_accuracy: 0.8500
Epoch 7/50
10/10 [==============================] - 34s 2s/step - loss: 0.2825 - accuracy: 0.8813 - val_loss: 0.3455 - val_accuracy: 0.8750
Epoch 8/50
10/10 [==============================] - 29s 1s/step - loss: 0.2568 - accuracy: 0.9031 - val_loss: 0.3465 - val_accuracy: 0.8750
```

Vgg16

Training with 5 Epochs - The model starts to learn basic patterns in the data With just 5 epochs, it's highly likely that the model will underfit, meaning it hasn't had enough time to learn the underlying patterns in the training data. The accuracy and loss metrics might show improvement, but the model's performance will likely be suboptimal.

Training with 50 Epochs - The model has more time to learn and adjust its weights, leading to better performance.With more epochs, there's a risk of overfitting, where the model learns the noise in the training data, reducing its performance on unseen data.Using techniques like early stopping helps mitigate overfitting by stopping training once the model performance on the validation set stops improving. More epochs usually result in higher training accuracy and potentially higher validation accuracy if overfitting is controlled.

ModelCheckpoint('best_inception_model.h5', monitor='val_accuracy', save_best_only=True): Saves the best model during training based on validation accuracy.
fit(train_gen, validation_data=val_gen, epochs=50, callbacks=[early_stopping, checkpoint]): Trains the model with the training data, validates it with validation data, and uses callbacks for early stopping and checkpointing.
Epochs = 5

```
Epoch 1/5
19/19 [==============================] - 252s 13s/step - loss: 0.5733 - accuracy: 0.7600 - val_loss: 0.3900 - val_accuracy: 0.7500
Epoch 2/5
19/19 [==============================] - 242s 13s/step - loss: 0.3328 - accuracy: 0.8533 - val_loss: 0.2884 - val_accuracy: 0.9100
Epoch 3/5
19/19 [==============================] - 244s 13s/step - loss: 0.2995 - accuracy: 0.8633 - val_loss: 0.3147 - val_accuracy: 0.8600
Epoch 4/5
19/19 [==============================] - 268s 14s/step - loss: 0.2677 - accuracy: 0.8833 - val_loss: 0.3668 - val_accuracy: 0.8200
Epoch 5/5
19/19 [==============================] - 241s 13s/step - loss: 0.2342 - accuracy: 0.9067 - val_loss: 0.3555 - val_accuracy: 0.8400
```

Epochs = 50

```
Epoch 4: val_loss did not improve from 0.35826
20/20 [==============================] - 226s 11s/step - loss: 0.2544 - accuracy: 0.8781 - val_loss: 0.3826 - val_accuracy: 0.8500
Epoch 5/50
20/20 [==============================] - ETA: 0s - loss: 0.2208 - accuracy: 0.8969
Epoch 5: val_loss did not improve from 0.35826
20/20 [==============================] - 211s 10s/step - loss: 0.2208 - accuracy: 0.8969 - val_loss: 0.4516 - val_accuracy: 0.8500
Epoch 6/50
20/20 [==============================] - ETA: 0s - loss: 0.2097 - accuracy: 0.9094
Epoch 6: val_loss did not improve from 0.35826
20/20 [==============================] - 209s 10s/step - loss: 0.2097 - accuracy: 0.9094 - val_loss: 0.3616 - val_accuracy: 0.9000
Epoch 7/50
20/20 [==============================] - ETA: 0s - loss: 0.2160 - accuracy: 0.9094
Epoch 7: val_loss did not improve from 0.35826
20/20 [==============================] - 208s 10s/step - loss: 0.2160 - accuracy: 0.9094 - val_loss: 0.5668 - val_accuracy: 0.7500
Epoch 8/50
20/20 [==============================] - ETA: 0s - loss: 0.2170 - accuracy: 0.9062
Epoch 8: val_loss did not improve from 0.35826
20/20 [==============================] - 214s 11s/step - loss: 0.2170 - accuracy: 0.9062 - val_loss: 0.4116 - val_accuracy: 0.8500
Model saved at: /content/repository/yiweichen04-retina_dataset-914b0f4/model/vgg16_model.h5
```

<table>
<tr>
<td>Efficientnet B1</td>
<td>

Learning Rate: Controls step size at each iteration.
Batch Size: Number of training samples per iteration.
Epochs: Number of complete passes through the training dataset.
Optimizer: Algorithm used to minimize the loss function (e.g., Adam,RMSprop).

GlobalAveragePooling2D: Reduces the spatial dimensions of the feature map by averaging, which helps in reducing the number of parameters and computational load.

Training with 5 Epochs - The model starts to learn basic patterns in the data With just 5 epochs, it's highly likely that the model will underfit, meaning it hasn't had enough time to learn the underlying patterns in the training data. The accuracy and loss metrics might show improvement, but the model's performance will likely be suboptimal.

Training with 50 Epochs - The model has more time to learn and adjust its weights, leading to better performance.With more epochs, there's a risk of overfitting, where the model learns the noise in the training data, reducing its performance on unseen data.Using techniques like early stopping helps mitigate overfitting by stopping training once the model performance on the validation set stops improving. More epochs usually result in higher training accuracy and potentially higher validation accuracy if overfitting is controlled.

```
#input layer
inputs=tf.keras.layers.Input(shape=(240,240,3),name='input layer')


#adding in data augmentation as a layer itself
x=data_augmentation(inputs)

#pass inputs to base model
x=base_model(x, training=False)
print(x.shape)

#We will use average pooling to reduce the size of the feature map.
x=tf.keras.layers.GlobalAveragePooling2D(name='global_average_pooling_layer')(x)
print(x.shape)

outputs=tf.keras.layers.Dense(1,activation='sigmoid',name='output_layer')(x)
model_0= tf.keras.Model(inputs,outputs)

#define early stopping callback.
early_stopping=EarlyStopping(monitor='val_accuracy',patience=3,restore_best_weights=True)
checkpoint = ModelCheckpoint('best_efficientnet_model.h5', monitor='val_accuracy', save_best_only=True)

model_0.compile(loss='binary_crossentropy',
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])

history_model_0= model_0.fit(train_data,
                             epochs=50,
                             validation_data=val_data,
                             callbacks=[early_stopping,checkpoint])
```

</td>
</tr>
</table>

Accuracy –

Epochs = 5

```
Epoch 1/5
10/10 [==============================] - 43s 2s/step - loss: 0.6160 - accuracy: 0.6875 - val_loss: 0.5550 - val_accuracy: 0.7500
Epoch 2/5
10/10 [==============================] - 31s 1s/step - loss: 0.4924 - accuracy: 0.7500 - val_loss: 0.4806 - val_accuracy: 0.7500
Epoch 3/5
10/10 [==============================] - 28s 1s/step - loss: 0.4315 - accuracy: 0.7937 - val_loss: 0.4196 - val_accuracy: 0.8000
Epoch 4/5
10/10 [==============================] - 29s 1s/step - loss: 0.3869 - accuracy: 0.8438 - val_loss: 0.3897 - val_accuracy: 0.8000
Epoch 5/5
10/10 [==============================] - 28s 1s/step - loss: 0.3470 - accuracy: 0.8594 - val_loss: 0.3798 - val_accuracy: 0.8000
```

Epochs = 50

```
Epoch 1/50
10/10 [==============================] - ETA: 0s - loss: 0.5833 - accuracy: 0.7031/usr/local/lib/python3.10/dist-packages/keras/src/e
  saving_api.save_model(
10/10 [==============================] - 49s 2s/step - loss: 0.5833 - accuracy: 0.7031 - val_loss: 0.5492 - val_accuracy: 0.7500
Epoch 2/50
10/10 [==============================] - 29s 1s/step - loss: 0.4924 - accuracy: 0.7594 - val_loss: 0.4763 - val_accuracy: 0.7500
Epoch 3/50
10/10 [==============================] - 31s 1s/step - loss: 0.4256 - accuracy: 0.8094 - val_loss: 0.4104 - val_accuracy: 0.8000
Epoch 4/50
10/10 [==============================] - 28s 1s/step - loss: 0.3824 - accuracy: 0.8562 - val_loss: 0.4005 - val_accuracy: 0.8000
Epoch 5/50
10/10 [==============================] - 27s 1s/step - loss: 0.3291 - accuracy: 0.8594 - val_loss: 0.3827 - val_accuracy: 0.8000
Epoch 6/50
10/10 [==============================] - 28s 1s/step - loss: 0.3217 - accuracy: 0.8719 - val_loss: 0.3631 - val_accuracy: 0.8250
Epoch 7/50
10/10 [==============================] - 27s 930ms/step - loss: 0.3006 - accuracy: 0.8813 - val_loss: 0.3517 - val_accuracy: 0.8250
Epoch 8/50
10/10 [==============================] - 26s 1s/step - loss: 0.2921 - accuracy: 0.8906 - val_loss: 0.3606 - val_accuracy: 0.8250
Epoch 9/50
10/10 [==============================] - 26s 1s/step - loss: 0.2749 - accuracy: 0.8875 - val_loss: 0.3477 - val_accuracy: 0.8250
```

Inception V3

rescale=1./255: Normalizes the image pixel values to the range [0,1]

flow_from_directory: Generates batches of data with real-time data augmentation.

target_size: Resizes the input images.
class_mode='categorical': Specifies that the target labels are one-hot encoded.
Flatten: Converts the 3D output of the base model to 1D.
Dense(2, activation='softmax'): Adds a fully connected layer with 2 output units and a softmax activation function for binary classification.

Training with 5 Epochs - The model starts to learn basic patterns in the data With just 5 epochs, it's highly likely that the model will underfit, meaning it hasn't had enough time to learn the underlying patterns in the training data. The accuracy and loss metrics might show improvement, but the model's performance will likely be suboptimal.

Training with 50 Epochs - The model has more time to learn and adjust its weights, leading to better performance.With more epochs, there's a risk of

	overfitting, where the model learns the noise in the training data, reducing its performance on unseen data.Using techniques like early stopping helps mitigate overfitting by stopping training once the model performance on the validation set stops improving. More epochs usually result in higher training accuracy and potentially higher validation accuracy if overfitting is controlled.

```
# Define Early Stopping and Model Checkpoint callbacks
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_inception_model.h5', monitor='val_accuracy', save_best_only=True)

# Train the model with early stopping and model checkpoint callbacks
history = inception_model.fit(train_gen, validation_data=val_gen, epochs=epochs, callbacks=[early_stopping, checkpoint])

# Save the model
model_save_path = '/content/best_inception_model.h5'
inception_model.save(model_save_path)
print(f"Model saved at: {model_save_path}")

# Evaluate the model on the test set
test_loss, test_accuracy = inception_model.evaluate(test_gen)
print(f"Test accuracy: {test_accuracy}")

# Predict the test data
predictions = inception_model.predict(test_gen)
y_pred = np.argmax(predictions, axis=1)
y_true = test_gen.classes

# Print the classification report
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=test_gen.class_indices.keys()))

# Print the confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_true, y_pred))
```

Accuracy –

Epochs = 5

```
Epoch 1/5
10/10 [==============================] - 35s 4s/step - loss: 0.2359 - accuracy: 0.8781 - val_loss: 616191.3125 - val_accuracy: 0.7500
Epoch 2/5
10/10 [==============================] - 31s 3s/step - loss: 0.1442 - accuracy: 0.9406 - val_loss: 25425.7383 - val_accuracy: 0.7500
Epoch 3/5
10/10 [==============================] - 33s 3s/step - loss: 0.1157 - accuracy: 0.9438 - val_loss: 1155.9987 - val_accuracy: 0.7500
Epoch 4/5
10/10 [==============================] - 32s 3s/step - loss: 0.0957 - accuracy: 0.9625 - val_loss: 41.6141 - val_accuracy: 0.7250
Epoch 5/5
10/10 [==============================] - 34s 3s/step - loss: 0.0715 - accuracy: 0.9656 - val_loss: 1.5806 - val_accuracy: 0.8000
```

Epochs = 50

```
Epoch 1/50
10/10 [==============================] - ETA: 0s - loss: 5.6772 - accuracy: 0.7219 /usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:
  saving_api.save_model(
10/10 [==============================] - 386s 36s/step - loss: 5.6772 - accuracy: 0.7219 - val_loss: 821.4874 - val_accuracy: 0.7500
Epoch 2/50
10/10 [==============================] - 411s 41s/step - loss: 1.6518 - accuracy: 0.8281 - val_loss: 152897.5312 - val_accuracy: 0.7500
Epoch 3/50
10/10 [==============================] - 420s 41s/step - loss: 0.4699 - accuracy: 0.8500 - val_loss: 19500656.0000 - val_accuracy: 0.7500
Epoch 4/50
10/10 [==============================] - 347s 34s/step - loss: 0.3396 - accuracy: 0.8656 - val_loss: 16894314.0000 - val_accuracy: 0.7500
Model saved at: /content/best_inception_model.h5
2/2 [==============================] - 13s 2s/step - loss: 878.6385 - accuracy: 0.7500
Test accuracy: 0.75
2/2 [==============================] - 14s 2s/step
```

Final Model Selection Justification :

Final Model	Reasoning
Resnet 50	As, Resnet50 and Vgg16 have a comparable train accuracy of 90%. But Resnet50 Validation accuracy is 87.5% and Vgg16 Validation accuracy is 85%